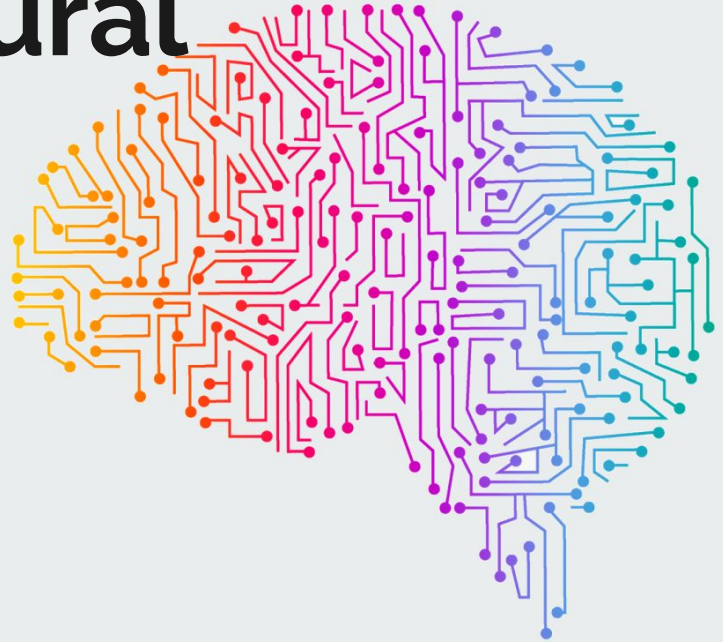




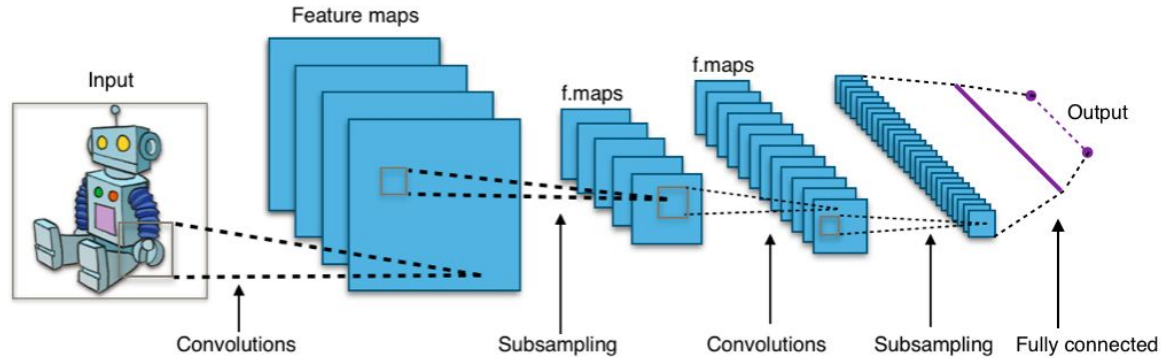
Convolutional Neural Network

Rita Folisi e Patrizio Palmisano



Richiami

Cosa sono?



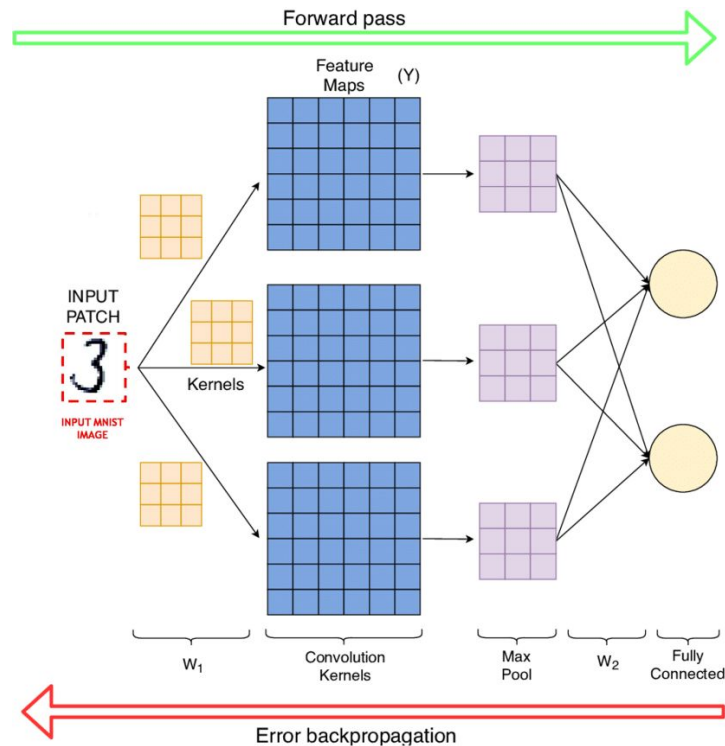
Obiettivo: insegnare ad una macchina come identificare e classificare un'immagine.

Idea: utilizzare molte immagini come training, alle quali vengono applicati filtri per estrarre feature.

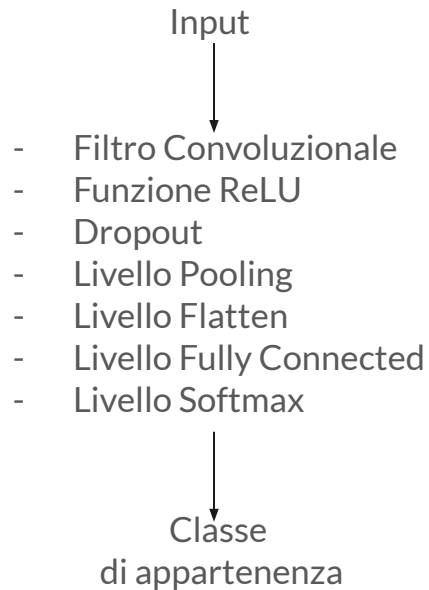
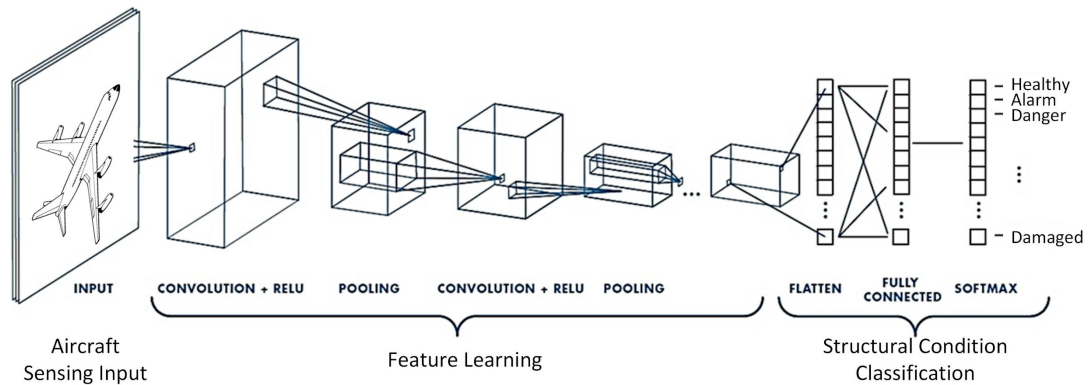
Queste, aggregate insieme, permettono il riconoscimento di pattern complessi.

Perché si usano?

- La macchina apprende quali pesi inserire nei filtri convoluzionali
- Si usa la **error backpropagation**
- Molto più comodo per il programmatore, che non dovrà precisare i parametri di ogni filtro



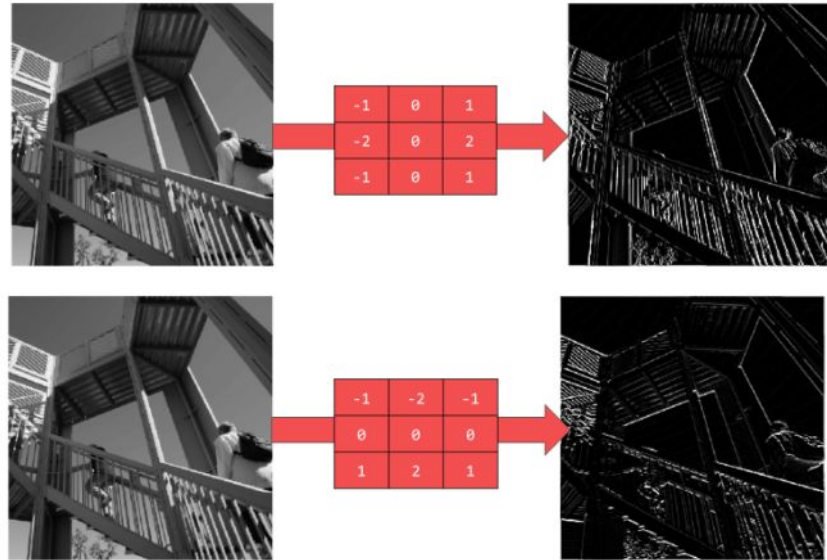
Come funzionano?




Filtri convoluzionali

Cosa sono?

- Un filtro in generale è una matrice di numeri
- Ogni filtro evidenzia una peculiarità dell'immagine
- Per immagini a colori, si usano filtri tridimensionali





1 <small>x1</small>	1 <small>x0</small>	1 <small>x1</small>	0	0
0 <small>x0</small>	1 <small>x1</small>	1 <small>x0</small>	1	0
0 <small>x1</small>	0 <small>x0</small>	1 <small>x1</small>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

Come funzionano?

- Su una immagine scorre il filtro di dimensione minore (stride)
- Per ciascuna posizione si effettua una moltiplicazione tra valori dell'input e pesi del filtro
- Il risultato viene salvato in Convolved Feature, che coglierà quindi una particolarità dell'immagine
- I pesi vengono appresi autonomamente
- Immagine ridotta (posso usare il **padding**)

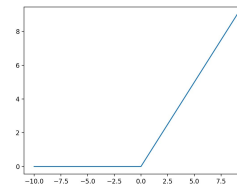
$$w_{\text{out}} = \left\lfloor \frac{w_{\text{in}} + 2p - k}{s} \right\rfloor + 1$$

ReLU

Funzione di attivazione che si occupa di azzerare i valori negativi.

Filter 1 Feature Map

9	3	5	-8
-6	2	-3	1
1	3	4	1
3	-4	5	1

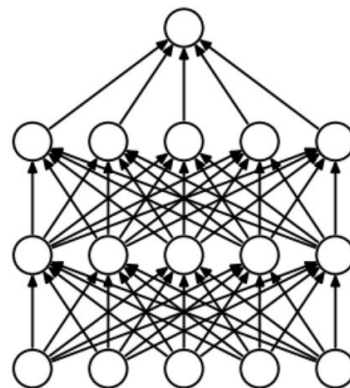


ReLU Layer

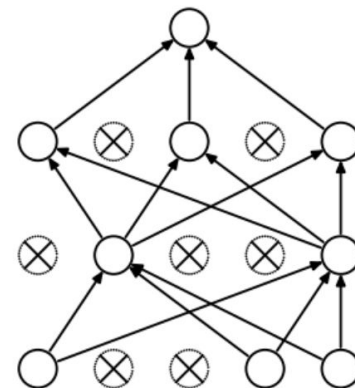
9	3	5	0
0	2	0	1
1	3	4	1
3	0	5	1

Dropout

Non considerare alcuni neuroni durante il forward o backward pass.
Utile per evitare overfitting.



(a) Standard Neural Net



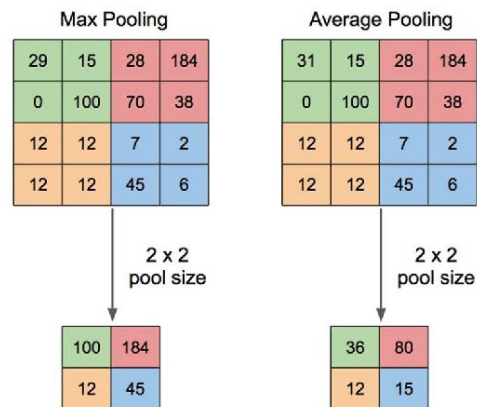
(b) After applying dropout.

Pooling

I pixel vengono raggruppati in un sottoinsieme.

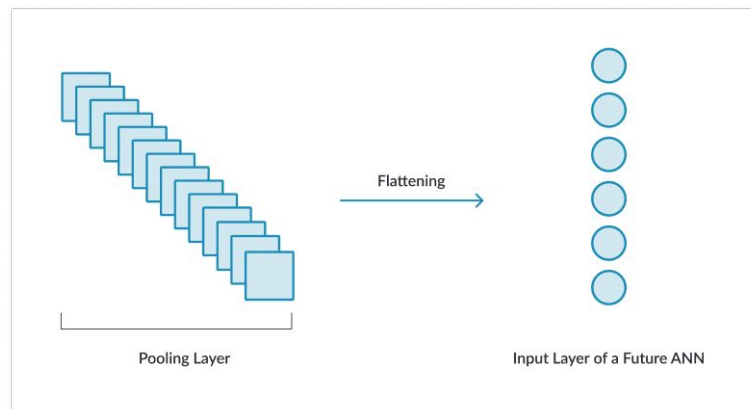
Diverse modalità, tra cui:

- Max Pooling
- Average Pooling



Flatten

Rende monodimensionali le activation maps.



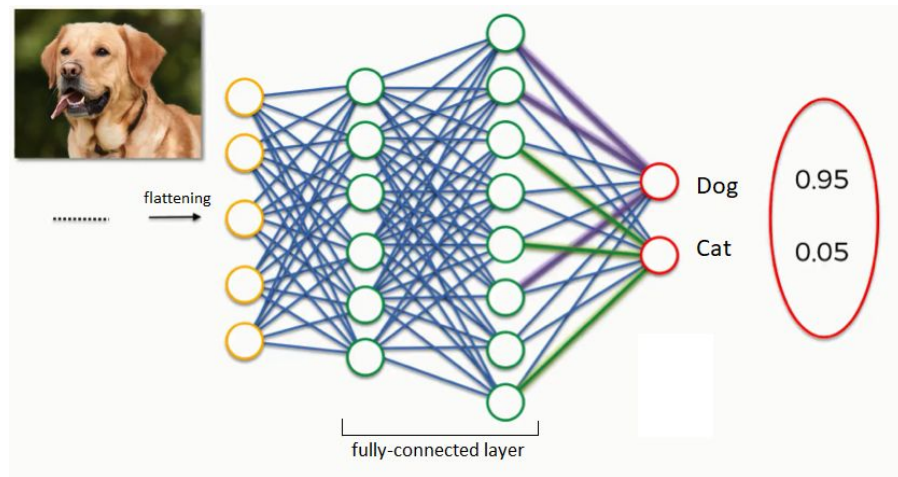
Fully connected

Idea: connettere tutti i neuroni di un layer con tutti i neuroni di un altro layer.

Si inizia ad effettuare la classificazione in base alle features estratte precedentemente.

Softmax

Genera la probabilità di appartenenza a tutte le classi proposte.



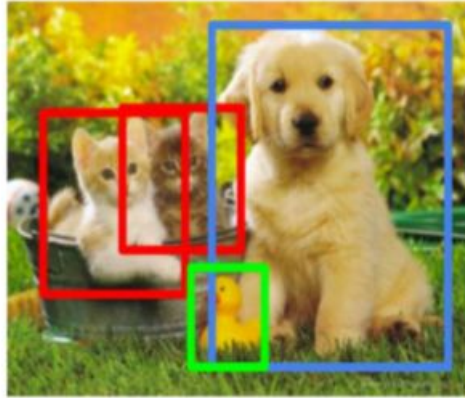
Altre applicazioni delle CNN

Semantic
Segmentation



CAT

Object
Detection



CAT, DOG, DUCK

Image
Captioning



The cat is in the grass.



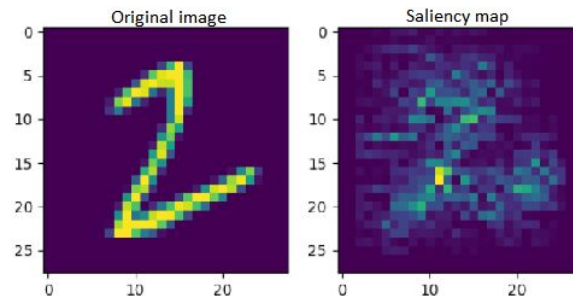
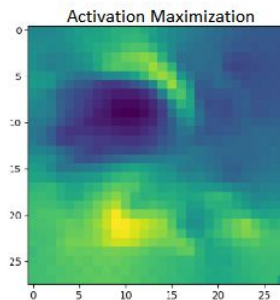
Passiamo al codice

Visualizzazione con le CNN

In generale, diversi tipi di visualizzazione:

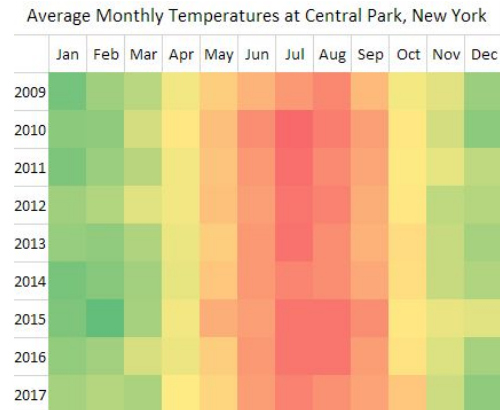
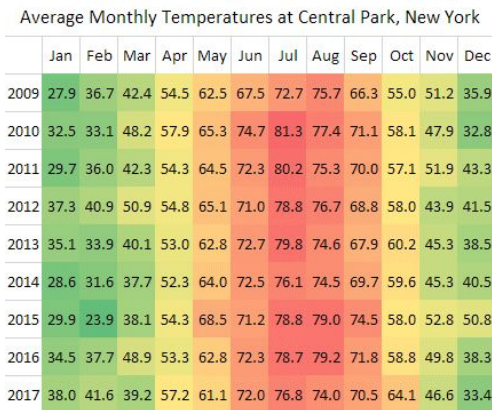
1. Heatmaps (matrice di confusione e Class Activation Maps)
2. Intermediate activation maps
3. Activation Maximations
4. Saliency Maps

Nell'esempio, le ultime due categorie in ordine



colori. Può avere molte applicazioni, tra queste vedremo:

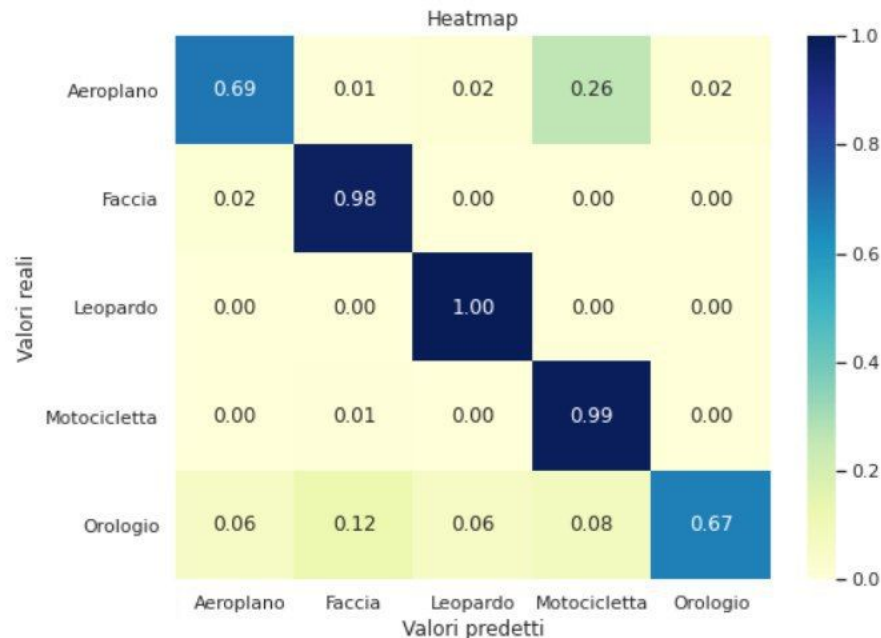
- associata alle matrici di confusione
- con le classi di attivazione (es: Grad CAM)



Heatmaps con matrice di confusione

Dal modello creato, è possibile generare la matrice di confusione, riproducibile come heatmap. Avrò:

- In ascissa i valori predetti
- In ordinata i valori reali
- Caselle più scure: più probabili
- Caselle più chiare: meno probabili





Small dataset: overfitting

Nell'heatmap nell'esempio MNIST l'accuratezza e gli errori sono più rari rispetto all'heatmap precedente.

Perché?

- Capienza del dataset MNIST: 60mila per training
- Capienza del dataset precedente: 2mila per training

Un **dataset ridotto** può causare un po' di overfitting e quindi comportare errori di classificazione

Come risolvere?

Aumento i filtri?

- **Data augmentation:** creo nuovo training data da materiale pre-esistente

```
train_datagen_a = ImageDataGenerator(  
    rescale=1./255,  
    rotation_range=40,  
    width_shift_range=0.2,  
    height_shift_range=0.2,  
    shear_range=0.2,  
    zoom_range=0.2,  
    horizontal_flip=True)
```



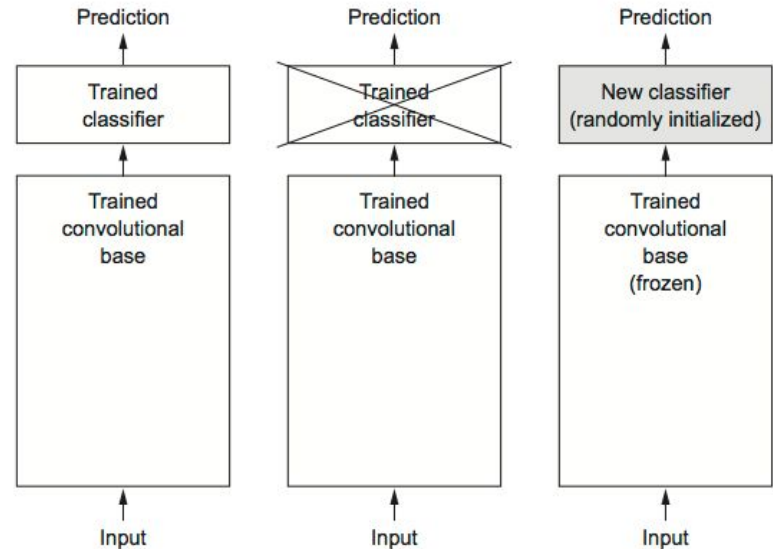
Data Augmentation



- Pre-trained model:

- Uso di un modello addestrato su un dataset molto esteso e variegato.
- Capacità di fornire informazioni utili con categorie non viste.
- Del modello, viene sostituito il classificatore con uno nuovo.
- La base convoluzionale rimane invariata.

```
conv_base.trainable = False
model = tf.keras.Sequential([
    conv_base,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation=tf.nn.relu),
    tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])
```

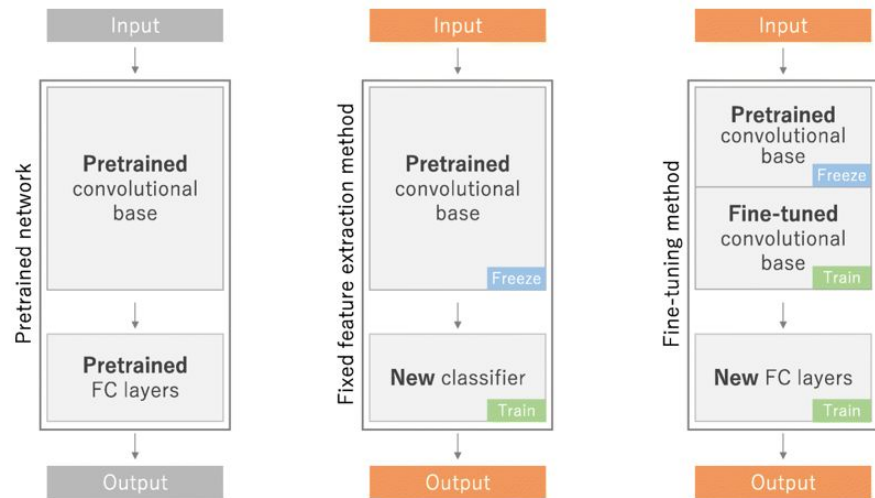


- Fine-tuning e pre-trained model:

- Si sbloccano alcuni strati della base convoluzionata freezata nel pre-trained model.
- Si addestrano il classificatore e gli strati sbloccati.

```
conv_base1.trainable = True

# We fine-tune the last convolutional block (Conv block 5)
for layer in conv_base1.layers:
    if layer.name == 'block5_conv1':
        layer.trainable = True
    else:
        layer.trainable = False
```



Intermediate activation maps

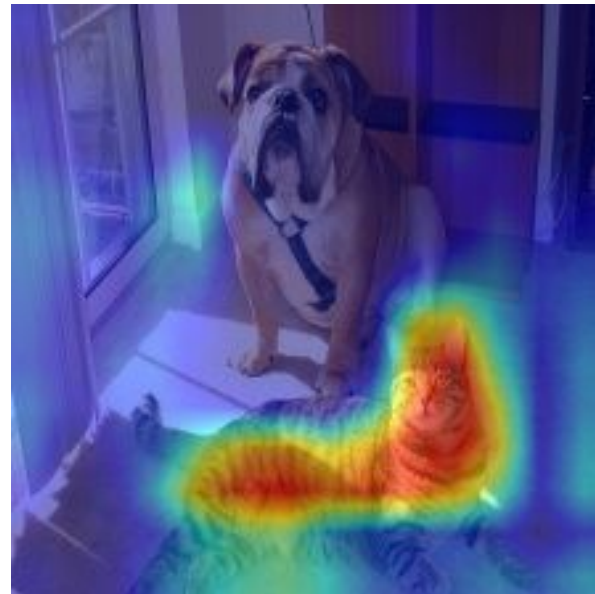
Activation: output ottenuto dopo aver applicato una activation function (filtro, pooling, etc).

Obiettivo: visualizzare come la rete rielabora l'immagine durante il forward pass e quali feature riesce a cogliere.



Grad CAM

- Visualizzazione di cosa sta guardando la neural network durante la classificazione, data una label
- Utile per vedere se la macchina sta lavorando correttamente.
- Uso del gradiente proveniente dall'ultimo layer
- Migliore localizzazione



Gatto



Ritorniamo al codice

